חלק ב':

- **partb.opt.s:** בקובץ זה נמצאו מספר טעויות:

  - בעת הגרלת המספר יש שימוש בפונקציה rand מ-libc שמגרילה מספר רנדומלי אל רגיסטר, שזו אמנם המטרה, אך היא מגרילה בכל פעם את אותו המספר (במקרה שלנו יצא 42). לכן, על מנת לפתור בעיה זו השתמשנו ב-srand. הפונקציה srand יוצרת seed לרנדום לפי הערך בראש המחסנית, ולכן קודם כל דחפנו את ערך החזרה של time הנמצא ב-eax לראש המחסנית כך שהרנדום יתבסס על זמן המערכת. לאחר מכן, ישנה הקריאה המקורית ל-rand שכעת תגריל בכל פעם מספר אחר.

  - בעת ריצת התוכנית, מודפסת פעמיים ההודעה "Guess a number between 1 and 100" כאשר בפעם השנייה אמורה להופיע ההודעה "Enter your guess:". לכן, על מנת לתקן זאת, החלפנו בשורה 39 בקובץ המקורי את LC0 ל-LC1 כך שתודפס ההודעה הרצויה.

  - לאחר קליטת הניחוש מהמשתמש, המספר שבחר מופיע במחסנית והמספר שהוגרל מופיע ברגיסטר ebx. בעת השוואת המספרים, בשורה 44 בקובץ המקורי, ניגשים אל המספר שהזין המשתמש ע"י ebp-12. אם המספר שניחש המשתמש קטן מהמספר שהוגרל, ישנה קפיצה אל L2 שגורמת לתחילת התהליך מהתחלה, זאת במקום להודיע על השגיאה בבחירת המספר והודעת בחירה מחדש. לכן, על מנת לתקן זאת, החלפנו בשורה 45 את jge L2 ל-jge L3 שתגרום להדפסת הודעה האומרת שהמספר שהמשתמש בחר קטן מדי ויש לבחור מחדש.

- **partb.nopt.s:** בקובץ זה נמצאו מספר טעויות:

  - בתחילת התכנית, בשורה 24, מקטינים את esp ב-16 כך שמשאירים מקום ל-4 משתנים/ ארגומנטים, למרות שבפועל משתמשים רק ב-3. לכן, הקטנו את esp ב-12 במקום.

  - המחרוזת הנמצאת ב-LC2 בשורה 11 מכילה c% ולא d%, ועל כן הקלט אינו מתבפרש כראוי (מתפרש כתו במקום כמספר שלם). לכן, שינינו זאת בהתאם.

  - בשורות 58 ו-65 (בקובץ המקורי), אליהן מגיעים כאשר המספר אותו הזין המשתמש הוא גדול מדי או קטן מדי, מודפסת הודעת שגיאה ולאחרי קופצים אל L3 שם מסתיימת ריצת התוכנית. על מנת לסדר זאת, יש להחליף את הקפיצה הנ"ל ולקפוץ אל L6 במקום, שם תתבצע קליטה מחדש של מספר ע"י המשתמש.

חלק ג':

להלן שלבי הפתרון בחלק זה:

- ראשית כל, נכנסנו לקוד המקור בכתובת המצורפת. לאחר סקירה מהירה ובחינת העצמים הנמצאים בקוד, הגענו לחלק שבו מופיע כפתור מוסתר שלחיצה עליו תביא לאתגר, זאת הסקנו ע"י הקוד:

  $class = "button\ disabled"\ \ onclick = "challenge\ me"$

  לכן, בעריכת קוד המקור מחקנו את "disabled" על מנת לאפשר את הכפתור וקיבלנו בתצוגה כפתור "Passwords Recovery". לחיצה על הכפתור חשפה תיאור של ה-challenge שעלינו לבצע עם אפשרות להעלאת קובץ exe.

- כתבנו קוד C הפותר את ה-challenge המבוקש. להלן הקוד המקורי:

```c
#include <stdio.h>
#define ROAD        "road"
#define SETTLEMENT  "settlement"
#define CITY        "city"
#define DEVELOPMENT "development"

enum CardType {
    WOOD,
    BRICK,
    WOOL,
    GRAIN,
    ORE,
    CardType_Count
};

int strcmpr(char *s1, char *s2)
{
    if ((s1 == NULL) || (s2 == NULL))
        return 1;
    while ((*s1 != '\0') && (*s2 != '\0') && (*s1 == *s2))
    {
        s1++;
        s2++;
    }
    return !(*s1 == *s2);
}

void chomp(char *s) {
    while(*s && *s != '\n' && *s != '\r') s++;

    *s = 0;
}

int main(int argc, char ** argv) {
    char buffer[255];

    int cardAmounts[CardType_Count];
    for (int i=0; i < CardType_Count; i++) {
        cardAmounts[i] = 0;
    }

    while (scanf("%s", buffer) != EOF) {
        chomp(buffer);

        if (strcmpr(ROAD, buffer) == 0){
            cardAmounts[WOOD]++;
            cardAmounts[BRICK]++;

        } else if (strcmpr(SETTLEMENT, buffer) == 0) {
            cardAmounts[WOOD]++;
            cardAmounts[BRICK]++;
            cardAmounts[WOOL]++;
            cardAmounts[GRAIN]++;

        } else if (strcmpr(CITY, buffer) == 0) {
            cardAmounts[GRAIN] += 2;
            cardAmounts[ORE] += 3;

        } else if (strcmpr(DEVELOPMENT, buffer) == 0) {
            cardAmounts[WOOL]++;
            cardAmounts[GRAIN]++;
            cardAmounts[ORE]++;
        }
    }

    for (int i=0; i < CardType_Count; i++) {
        char* zeroPad = "";
        if (cardAmounts[i] < 10) {
            zeroPad = "0";
        }

        printf("%s%d ", zeroPad, cardAmounts[i]);
    }

}
```

לאחר קומפילציה, קיבלנו קובץ executable במשקל 103KB. עם נסיון העלאתו, קיבלנו הודעת שגיאה כי יש להעלות קובץ שגודלו לא עולה על 2KB.

- המרנו את הקוד הנ"ל לאסמבלי ע"י godbolt.org. נעזרנו בקוד שראינו בתרגול לביצוע הדפסה ע"י מציאת הפונקציות ויישמנו זאת עבור printf ו-scanf. התהליך כולו כלל את טעינת msvcrt.dll בצורה דינאמית ע"י LoadLibraryA, חיפוש כל אחת מהפונקציות ב-dll ע"י GetProcAddress שאותה מצאנו באמצעות הפונקציה search_kernel32. את הפונקציה search_kernel32, לה קראנו בקוד FindFunction, קיבלנו באתר הקורס כבר כ-hexa ולכן רק השארנו לה label תואם בסוף האסמבלי. עבור כל אחת מהפונקציות הנ"ל, יצרנו שגרות GetPrintF ו-GetScanF המוצאות את הכתובת לפונקציה ומחזירות אותה על גבי הרגיסטר eax.

- שתלנו בקוד האמסבלי שלנו את הפונקציות שיצרנו והחלפנו כל קריאה סטנדרטית ל-scanf ול-printf בקריאה לפונקציות היעודיות שיצרנו. בנוסף, הסרנו כל שימוש במחרוזות ובמקום דחפנו את הקידוד שלהן על המחסנית.

- המרנו את האסמבלי שקיבלנו ל-hexa ע"י שימוש באסמבלר של defuse ועברנו ליצירת קובץ PE. ראשית, נשים לב כי לא השתמשנו בפונקציות חיצוניות בצורה סטטית ולכן אין טבלת import, ואיננו מייצאים פונקציות ועל כן אין טבלת export. בנוסף, לא השתמשנו במחרוזות ודחפנו במקום על המחסנית את קידודן ועל כן גם אין לנו data section. כלומר, כל שיופיע הוא ה-header וה-text section שמכיל את הקוד, בדומה למה שראינו בתרגול. לכן, השתמשנו ב-header שהופיע בתרגול היות ואין צורך לשנות בו כלום, ואילו רק בטבלת ה-sections עלינו לשנות את המידע אודות ה-text section שלנו המכיל את הקוד. לכן, רצינו להגיע לתחילת ה-section. שמו הוא "text.", שזה ב-hexa מתורגם כ-"2E74657874" ולכן מצאנו ערך זה וקפצנו ישירות אליו (ב-PE הקיים מהדוגמה שראינו). הערכים הבאים מוגדרים כ-DWORD, כלומר מילה כפולה – 32 ביט, ולכן נעבור עליהם לפי כפולות של 8 ערכי hexa. בפועל, הערכים היחידים שעלינו לשנות הם הערכים המתארים את גודל ה-section, כלומר גודל הקוד, שהם VirtualSize ו-SizeOfRawData, שהם הערכים הראשון והשלישי בהתאמה לאחר שם ה-section. גודל הקוד שלנו הוא 853B וגודל הפונקציה search_kernel32 הוא 146B, כלומר ביחד הם 999B. בייצוג hexa של שני בתים זה מתפרש כ-0x03e7 וכמו שראינו בתרגול הבית העליון הוא הימני והתחתון הוא השמאלי ולכן הפכנו את הבתים כך שהגודל הוא e703. שינינו את שני הגדלים ב-header לגודל שהתקבל, והשתמשנו ב-hexed.it שבו שתלנו את ה-header החדש, אחריו את הקוד שלנו ואחריו את הפונקציה search_kernel. ייצאנו לקובץ challenge.exe שגודלו 2KB והעלינו אותו לאתר.

- הגענו לאתגר הבא, ובו לאחר לחיצה על Download Recovery Source הורדנו את הקובץ crackme.S. מניתוח דינאמי וניתוח סטטי של הקובץ ניתן להבין כי תתקבל הודעת הצלחה בשלב הראשון אם יתקבל ארגומנט אחד ומעלה, ללא תלות בערכו או בקלט. לכן, על מנת לעבור את השלב הראשון נריץ את הקובץ עם ארגומנט 1.

- בשלב השני, מניתוח סטטי של הקוד ניתן לראות מספר מחרוזות (בדיוק תשע) שנטענו על המחסנית החל מ-ebp-38 ועד ל-ebp-46. לאחר מכן, מוגרלים תשעה מספרים ומוכנסים בזה אחר זה על גבי המחסנית, החל מ-ebp-37 ועד ל-ebp-29. לאחר מכן, ניתן לראות כי מתקבלים בקלט שני מספרים הנטענים ל-ebp-24 ו-ebp-28. מתבצע וידוא כי מתקבל קודם מספר הגדול מאפס ולאחריו מתקבל מספר הגדול ממנו. עבור שני המספרים (שנסמנם a,b)

מתבצעת הדפסה של תוכן המחסנית החל מ-ebp-37+a עד ל-ebp-37+b, כאשר כמו שאמרנו ebp-37 הוא הערך שהחל ממנו ומטה הוזנו למחסנית תשעה מספרים אקראיים.
לאחר מכן, החל מהכתובת ebp-37-a, בכל פעם קולטים קלט הקסה-דצימלי של עד 8 בתים, מבצעים לו xor עם הערך הראשון שהודפס (ebp-37-a), ודוחפים את התוצאה למחסנית במקום הערך הקודם איתו ביצענו xor. מבצעים זאת בצורה איטרטיבית לכל אורך הטווח שהזנו (בין ebp-37+a עד ebp-37+b) וכך מחליפים את כל הערכים בטווח זה על המחסנית בערך ה-xor שלהם עם הקלטים שהתקבלו.
לאחר מכן, עוברים על המחרוזות הקבועות שעל המחסנית לפי הסדר, החל מ-ebp-46 ובמקביל עוברים על תוצאות ה-xor-ים שעל המחסנית החל מ-ebp-37 באמצעות counter, כאשר בכל פעם משווים בין התוצאות. אם כל התוצאות שוות, נקבל הצלחה בשלב 2.
לכן, ראשית נבנה את הקלט הראשוני כך שיודפסו לנו כלל ערכים אלה מהמחסנית ועל כן נזין את הערכים 0 ו-10, כך שיודפסו לנו הערכים ב-3 הכתובות הראשונות ובהם כל התוצאות.
לאחריהם נבנה קלטים כך שבהפעלת xor שלהם עם המחרוזות הקבועות נקבל את הערכים שהוגרלו. אנו יודעים כי אם a xor b = c אז b xor c = a ולכן נבצע xor של המחרוזות הקבועות שעל המחסנית עם הערכים שהודפסו לנו לפי סדר הבדיקה שלהם וכך נקבל מחרוזות מסוימות, שבעת הפעלת xor שלהן עם המחרוזות הקבועות על המחסנית נקבל את ערכי המחרוזות שהוגרלו.
הערכים שקיבלנו שיש להזין הם:

a5000000
e8687c1a
1ee60f04

ועבורם נקבל הצלחה בשלב 2.

- לאחר היציאה מהפונקציה המבצעת את שלב 2, ניתן לשים לב מניתוח סטטי כי הקוד אינו מגיע לאיזור שבו מודפסת המחרוזת המתאימה לסיום שלב 3. לכן, הלכנו בשיטה ההפוכה ומצאנו את המחרוזת וראינו כי היא מודפסת בפונקציה dummy שאליה אין לנו קפיצה. לכן, עלינו לגרום לכך שנגיע לפונקציה זו. החלטנו להשתמש בדריסת הערכים על המחרוזות שבוצעה בשלב הקודם ע"י חישוב ה-xor-ים של הקלט עם המחרוזות שהודפסו קודם לכן (עם הזנת הטווח להדפסה). לכן, כל שעלינו לעשות הוא לתת טווח מספיק גדול להדפסה כך שנראה את כתובת החזרה מהפונקציה מה-main ואת הכתובת של dummy הנמצאת על המחסנית, והזנת קלט שיעבור את שלב 2 בתוספת דריסת כתובת החזרה. מניתוח המחסנית בריצה עם IDA, ניתן לראות כי כתובת החזרה נמצאת כצפוי ב-ebp+4 (היות והיא נדחפת למחסנית לפני החלפת המחסניות) והכתובת של dummy נמצאת שתי כתובות מתחתיה. ניתן לראות זאת בנוסף ע"י הזנת טווח של 0 עד 72 וכך הכתובת של dummy תהיה הערך השלישי מהסוף שיודפס לנו וכתובת החזרה תהיה הכתובת האחרונה. כעת, כדי לעבור את שלב 2 עלינו להשאיר את שלושת הערכים הראשונים זהים לערכם הקודם על מנת שבדיקת ה-xor-ים תעבור בהצלחה ונצליח את שלב 2, לאחר מכן נרצה לשתול אפסים שלא ישנו את יתר הערכים פרט לכתובת החזרה (14 פעמים 0x00000000) ואחריהם נגיע לכתובת החזרה. ניקח את הכתובת של dummy ונחשב xor בינה לבין כתובת החזרה, ואת התוצאה ניתן כקלט כך שבחישוב xor של קלט זה עם כתובת החזרה הקיימת נקבל את הכתובת של dummy. בהרצה על מחשבינו נראה כי הכתובת של dummy היא 0x4018c7 וערך ה-xor שלה עם כתובת החזרה הוא 0xb25 ולכן את ערך זה נזין במקום התואם לכתובת החזרה וכך יגרום לקפיצה אל dummy. מהרצה באתר כתובת החזרה היא 0x401236 והכתובת של

dummy היא 0x401767 שערך ה-xor ביניהן הוא 551 ולכן הקלט בהתאמה וקיבלנו את ההודעה על הצלחת שלב 3.

- כעת, כמו בשלב הקודם, אנו רואים שה-flow של הקוד לא יקח אותנו אל השלב בו מודפסת הצלחה בשלב 4 ומחפשים אותו ידנית. אנו רואים כי ההדפסה מתבצעת ע"י הפונקציה handler, שמופיע בתחילת הקוד כארגומנט לפונקציה signal יחד עם ארגומנט נוסף – 8. מחקירת הפונקציה מצאנו כי signal היא פונקציה המגדירה שגרת טיפול בפסיקות, כאשר 8 הוא מספר הפסיקה המייצג Floating Point Exception והכתובת של handler שהועברה היא שגרת הטיפול ל-FP. ממעבר על הקוד של handler, ניתן לשים לב כי הדבר הראשון שמתבצע הוא בדיקת הארגומנט המועבר בתור קוד השגיאה והשוואתו ל-8. אם הארגומנט אינו 8, מתבצעת יציאה מהפונקציה ללא ההדפסה הרצויה. לכן, על מנת להגיע ל-handler עלינו לגרום לחריגת FP כך שיועבר מספר השגיאה 8 ואכן תתרחש ההדפסה. עברנו על הקוד במטרה למצוא פעולות חילוק וראינו כי ב-dummy (אליה כבר הגענו בשלב 3) מבצעים חילוק של המספר 236496 (מקורי!) בערך הנמצא ב-divider. הערך המוזן אל divider מועבר מתוך eax בסוף ה-main, ולכן עלינו לדאוג כי הערך שיתקבל ב-eax בסוף הלוגיקה הקיימת שם יהיה 0. מניתוח הקוד ניתן לראות כי הערך ברגיסטר eax תלוי בארגומנט שקובץ ההרצה מקבל ועבור השמת הארגומנט 2- יתקבל בסוף ה-main ערך 0 ל-eax, שיושם ב-divider. לכן, הוספנו את הארגומנט -2. בעת סיום ריצת ה-main, כתובת החזרה תקפוץ ל-dummy כמו שהוגדר בסעיף הקודם ושם תתבצע החלוקה ב-0 שתגרום להדפסת הודעת ההצלחה. נציין כי עם ארגומנט יחיד נוצרת לולאה אינסופית בעת הרצת הקלט שלנו באתר, ולכן בחרנו להזין את -2 כארגומנט נוסף. עם זאת, הדבר שינה את שלוש המחרוזות הראשונות המודפסות מהמחסנית ועל כן התאמנו את התשובה לסעיף 2, כך שהתשובה המעודכנת היא:

a9000000
ec824063
e5379c91

- ולבסוף, שמנו לב שלאחר הקפיצה ל-dummy משלב 3 ישנה קריאה לפונקציה בשם db_access שבתוכה יש קריאות לפונקציות של sqllite3. תחילה open, אז exec ו-close. הנחנו שבשלב זה מבקשים את נתוני המשתמש מהDB. עקבנו אחר פונקציית exec שמקבלת מצביע לDB ומחרוזת שאילתה, כארגומנט השאילתה מעבירים שרשור של מחרוזת קבועה והארגומנט השני של התוכנית (שמועבר ע"י פונקציית main לתווית arg מוקדם יותר) בנוסף השרשור מתבצע ע"י strncat כך שיש הגבלה על מספר התווים שניתן לשרשר, 11 תווים במקרה הזה. לאחר השרשור מוסיפים בקצה המחרוזת את התו " ' ". כך שלבסוף תתקבל המחרוזת הבאה כשאילתה:

```
"select username, password from users where username=' ARGUMENT2 ' "
```

מובן שאין בסיס נתונים במחשבים האישיים אז התחלנו להריץ ניסויים להתקפת SQL injection באתר, ולבסוף המחרוזת הבאה חשפה את נתוני כל המשתמשים בבסיס הנתונים:

```
" '  OR 1=1 -- "
```

כאשר ה-' הראשון סוגר את התנאי הראשון, כלומר, שם-משתמש ולאחריו משנים את התנאי ל-or עם תנאי שמתקיים תמיד, 1==1 (בשפת SQL עם = יחיד). לכן התנאי הכולל מתקיים תמיד והשאילתה מחזירה את כל הצמדים (משתמש, סיסמה) העונים על התנאי, שהם כל הצמדים, קרי, כל המשתמשים וסיסמאותיהם. כדי להתעלם מה-' שהתוכנית מוסיפה בסוף המחרוזת הוספנו סימן הערה "—". לבסוף נתקבלה השאילתה:

```
" select username, password from users where username='' or 1=1 --  ' "
```